3D Printing Spatially Varying Color and Translucency
Supplemental Appendix

ALAN BRUNTON, CAN ATES ARIKAN, and TEJAS MADAN TANKSALE, Fraunhofer IGD
PHILIPP URBAN, Fraunhofer IGD and Norwegian University of Science and Technology NTNU

CCS Concepts: • **Computing methodologies → Perception**; *Graphics file formats*;

Additional Key Words and Phrases: 3D printing, metamerism, translucency, distance field

ACM Reference Format:
Alan Brunton, Can Ates Arikan, Tejas Madan Tanksale, and Philipp Urban. 2018. 3D Printing Spatially Varying Color and Translucency Supplemental Appendix. *ACM Trans. Graph.* XX, XX (August 2018), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

A RENDERING EQUATION

The radiance $\mathbf{L}_\lambda$ emitted from the object for an irradiance $\mathbf{E}_\lambda$ defined at each surface location $\vec{x}_i \in \partial\mathcal{S}$ is given by [Nicodemus et al. 1977]

$$\mathbf{L}_\lambda(\vec{\omega}_r, \vec{x}_r) = \int_{\partial\mathcal{S}} \int_{\Omega} \mathbf{B}_\lambda(\vec{\omega}_i, \vec{x}_i, \vec{\omega}_r, \vec{x}_r) d\mathbf{E}_\lambda(\vec{\omega}_i, \vec{x}_i) d\vec{\omega}_i d\vec{x}_i \qquad \text{(S1)}$$

where $\partial\mathcal{S}$ is the surface of the object and Ω is the unit hemisphere. Self-shadowing is ignored for simplicity.

B PROOFS RELATED DISTANCE COMPUTATION

Proof of Claim 1. The 2D squared Euclidean distance transform of a function $f(x,y)$ computes the following [Felzenzwalb and Huttenlocher 2012]

$$DT(f)(x,y) = \min_{x',y'} \left(f(x',y') + (x'-x)^2 + (y'-y)^2 \right) \qquad \text{(S2)}$$

If we apply step (2) only to a single (x,y), denoted by (x_0,y_0), it is equivalent to ignoring the surface voxels for all other $(x,y) \neq (x_0,y_0)$, as if the only surface voxels were those with coordinates (x_0,y_0,z). The distance transform would then compute for each (x,y)

$$\begin{aligned} d_s(x,y) &= d_s(x_0,y_0) + (x_0-x)^2 + (y_0-y)^2 \\ &= \min_{z \in \mathbf{Z}_i(x_0,y_0)} (z-z_s)^2 + (x_0-x)^2 + (y_0-y)^2 \end{aligned} \qquad \text{(S3)}$$

which is the squared Euclidean distance to the nearest surface voxel with coordinates (x_0,y_0,z) since all other values in $\mathbf{Z}_i(x_0,y_0)$ would increase it. When we apply step (2) to all (x,y), then (S2) finds the (x_0,y_0), which minimizes (S3). This is essentially the same argument why the separability of the squared Euclidean distance allows the distance transform in 2D to be computed as a sequence of 1D distance transforms.

The test in (2b) ensures that the squared z distance is to the correct surface $\partial\mathcal{S}_i$ s.t. $\text{id}_i = o_s(x,y)$. The use of independent 1D DTs for each id_i in step (3) ensures that only voxels (x',y') s.t. $o_s(x',y') = o_s(x,y)$ are considered in (S2). The use of the algebraic lower envelope representation [Felzenzwalb and Huttenlocher 2012] ensures that the distance is computed as the intersection of parabolas, independent of the presence of voxels $o_s(x',y) \neq o_s(x,y)$ (resp. $o_s(x,y') \neq o_s(x,y)$). □

Proof of Claim 2. Step (1) is trivially $O(1)$ per voxel. Step (3) runs in $O(WH)$ [Felzenzwalb and Huttenlocher 2012], which is $O(1)$ per voxel since there are WH voxels per slice. This is true even for the multi-track case, in each 1D DT a voxel contributes to the construction of only one lower envelope, and its distance is computed from only one lower envelope. Both these steps require only a single slice to be stored, plus $O(\max(W,H))$ for the multi-track DT. When processing slices in ascending order of z_s, the index $k(x,y)$ can be updated by comparing $\mathbf{Z}_i(x,y)[k(x,y)]$ and $\mathbf{Z}_i(x,y)[k(x,y)+1]$ (with appropriate bounds checking), which is $O(1)$ per non-empty $\mathbf{Z}_i(x,y)$. Since each voxel can belong to at most one object, the total number of non-empty lists, $\sum_{i=1}^{n} |\{\mathbf{Z}_i(x,y) \neq \emptyset \ \forall \ (x,y)\}| = O(WH)$. Thus, step (2) is $O(1)$ per voxel. This requires additional storage of an index for every non-empty $\mathbf{Z}_i(x,y)$, which is $O(WH)$. The storage of $\mathbf{Z}_i(x,y)$ and $\mathbf{RGBA}_i(x,y) \ \forall \ (x,y)$ is $\ll WHD$ since this grows linearly w.r.t. the surface area of $\partial\mathcal{S}_i$. □

C PROCEDURE TO GENERATE LOOK-UP TABLE

We use the following procedure to generate the color and translucency lookup table described in Sec. 5.4.

(1) We start with a regular grid with N grid points per dimension spanning with it's nodes $\mathbf{x}_i \in \mathbf{CIELAB}\alpha,\ i \in \{1,\ldots,N\}^4$, the whole $\mathbf{CIELAB}\alpha$ space.
(2) The colors of the nodes are mapped into $\mathbb{G}_0 \subset \mathbf{CIELAB}$, the media relative color gamut belonging to $\gamma = 0$:

$$\mathbf{y}_i \quad = \quad \mathbf{F}(\mathbf{x}_i) \in \mathbb{G}_0 \times \alpha \subset \mathbf{CIELAB}\alpha, \qquad \text{(S4)}$$

where $\mathbf{F} : \mathbf{CIELAB}\alpha \mapsto \mathbb{G}_0 \times \alpha$ is a point-wise gamut mapping method [Morovič 2008] for the color dimensions and the identity for α. We use a hue preserving, CUSP clipping algorithm in a hue-linearized CIELAB color space.
(3) For a sample set $\mathbb{S} \subset \mathbf{CMYK}\gamma$, we compute $\mathbf{h}(\mathbb{S}) \subset \mathbf{CIELAB}\alpha$. $\mathbb{S}$ can be defined by an 8-bit quantization of $\mathbf{CMYK}\gamma$. We assign a set of $\mathbf{CMYK}\gamma$ values to each node i as follows

$$\mathbb{H}_i \quad := \quad \{\mathbf{a} \in \mathbb{S} \mid [\mathbf{y}_i]_{\mathbf{CIELAB}} = [\mathbf{h}(\mathbf{a})]_{\mathbf{CIELAB}}\} \qquad \text{(S5)}$$

Authors' addresses: Alan Brunton, alan.brunton@igd.fraunhofer.de; Can Ates Arikan, arikan.canates@googlemail.com; Tejas Madan Tanksale, tejas.madan.tanksale@igd.fraunhofer.de Fraunhofer IGD; Philipp Urban, philipp.urban@igd.fraunhofer.de Fraunhofer IGD Norwegian University of Science and Technology NTNU.

0730-0301/2018/8-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

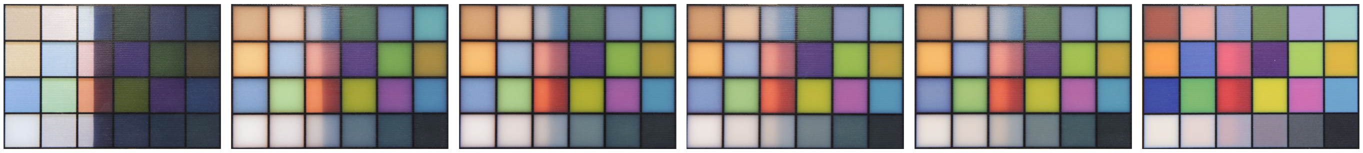

Fig. S1. Color checkers with varying degree of translucency backlit with a step function. From left to right: $\alpha = 0.201$ (maximally transparent), $\alpha = 0.348$, $\alpha = 0.493$, $\alpha = 0.640$, $\alpha = 0.769$, $\alpha = 0.786$ (maximally opaque).

where $[\cdot]_{\mathbf{CIELAB}} : \mathbf{CIELAB}\alpha \mapsto \mathbf{CIELAB}$ is an operator that returns the color dimensions rounded each to the closest integer. Note that the rounding error in **CIELAB** is below the just noticeable distance. $\mathbb{H}_i$ can be interpreted as the set of printer control values resulting in translucency metamers to $\mathbf{y}_i$.

(4) We compute an initial lookup table $\mathbf{p}_o$ from these metamer sets as follows

$$\mathbf{p}_o(\mathbf{x}_i) \quad := \quad \underset{\mathbf{a}\in\mathbb{H}_i}{\operatorname{argmin}} \, |\mathbf{x}_i - \mathbf{h}(\mathbf{a})|_\alpha \tag{S6}$$

where $|\cdot|_\alpha : \mathbf{CIELAB}\alpha \mapsto [0,1]$ returns the absolute value in the α-dimension. In the case that the optimization problem (S6) has multiple solutions, we can use standard separation strategies from printing, such as *Gray Component Replacement* (GCR) or *Under Color Removal* (UCR), to select a solution with distinct properties. We use a minimum GCR strategy; we replace the minimum amount of CMY with black, creating low graininess at the cost of reduced color constancy.

(5) Finally, we iteratively smooth the node values to ensure artifact-free reproductions of color and translucency transitions in textures, ensuring color accuracy: In iteration $j+1$, the component-wise mean $\mathbf{m}_i \in \mathbf{CMYK}\gamma$ from the $3\times3\times3\times3$ window centered at node i is computed using node entries of iteration j. Then, the new node value is assigned as follows:

$$\mathbf{p}_{j+1}(\mathbf{x}_i) := \underset{\mathbf{a}\in\mathbb{H}_i}{\operatorname{argmin}} \, \|\mathbf{m}_i - \mathbf{a}\|_2 \tag{S7}$$

We observed that this converges after a few dozen iterations. Remaining outliers are replaced by the mean value.

D ADDITIONAL RESULTS

Fig. S1 shows the CCs lit from behind with a step function illuminant. Cropped views of these images are shown in Fig. 8 (top). Fig. S2 (a) shows a model holding a beer, lit from the side. The model is printed maximally opaque, but the glass and beer are printed maximally transparent. The changes in alpha are created by editing the texture in a standard image editor. Fig. S2 (b) shows a close-up of the arm of the anatomy model in Fig. 10. Fig. S2 (c,d)show the same anatomy model with a different shader, creating translucency gradients rather than a sharp edge. Further, the color of the skin is kept, rather than changing it to RGB= $(1,1,1)$. Consider the difference between (b) and (d). Note that the gradient begins not at maximally opaque, but at relative $\alpha = 0.54$.

Fig. S3 and S4 show measured materials assigned to arbitrary geometries in a spatially varying way, mixing both measurements and design. Fig. S3 shows the St. Lucy model with measured RGBA values corresponding to violet stone (top), green stone (middle) and green soap (bottom). RGBA values are linearly interpolated in between. Fig. S4 shows the bunny model RGBA measured from the salmon sample at right. The white stripes are added procedurally.

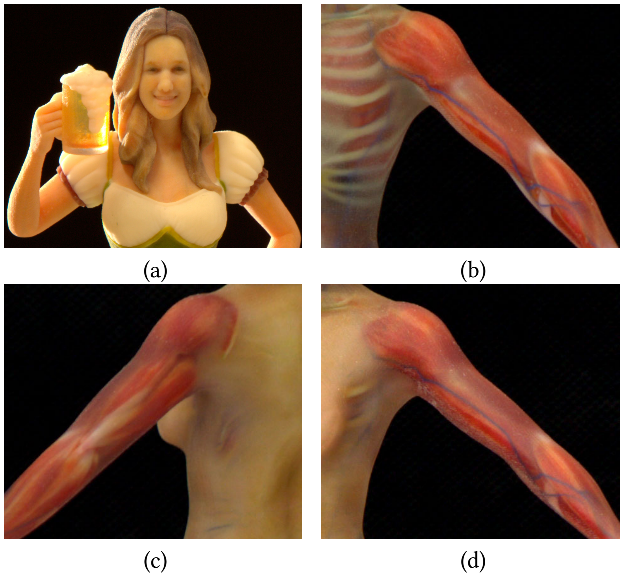

Fig. S2. Various design examples. See text for explanation

REFERENCES

P.F. Felzenzwalb and D.P. Huttenlocher. 2012. Distance Transforms of Sampled Functions. *Theory of Computing* 8, 19 (2012), 415–428.

J. Morovič. 2008. *Color Gamut Mapping.* John Wiley & Sons.

F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. 1977. *Geometrical considerations and nomenclature for reflectance.* United States. National Bureau of Standards.

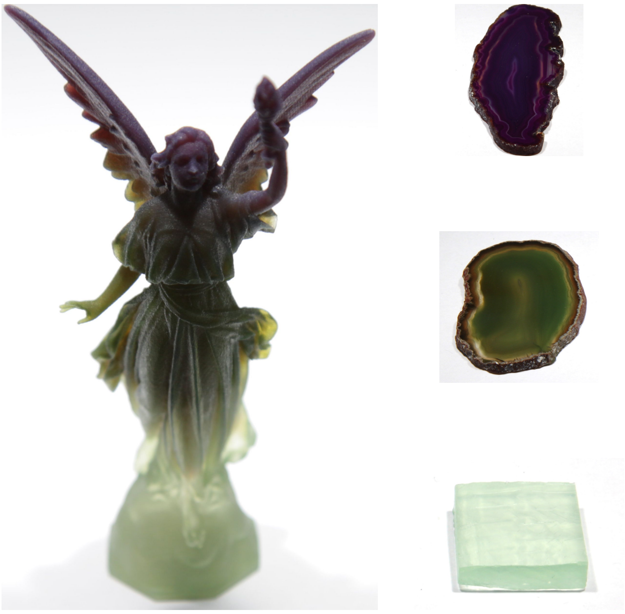

Fig. S3. The St. Lucy model printed (10cm) with varying RGBA values. At the top is that of the violet stone, middle is green stone and bottom is green soap, with linear transitions in between.

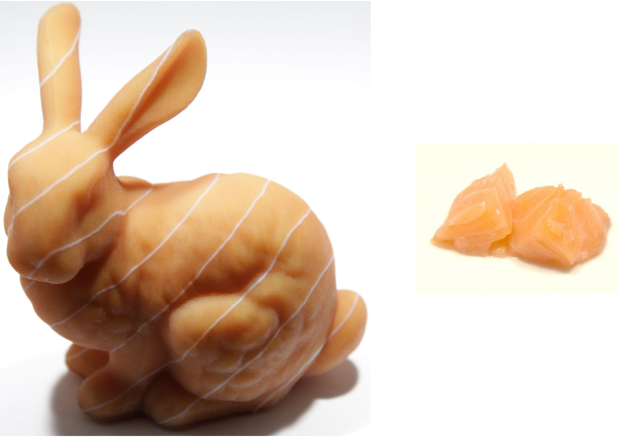

Fig. S4. The Stanford Bunny printed (10cm) with the measured RGBA values from the salmon. The white, opaque stripes are added procedurally.